
cepton_sdk Documentation

Cepton Technologies

Sep 12, 2019

CONTENTS

1	Overview	1
2	Errors	3
3	Setup	5
4	General	7
5	Sensors	9
6	Points	11
7	Serial	13
8	Capture Replay	15
9	Export	17
10	Samples	19
	Index	23

OVERVIEW

If a method is undocumented, consult the *C/C++ SDK* documentation, since many methods in this library are just wrapper functions.

1.1 Timestamps

Unless otherwise marked, all timestamps are seconds since the Unix epoch (UTC). Note that this differs from the *C/C++* interface which uses microseconds.

ERRORS

```
class cepton_sdk.C_ErrorCode
    An enumeration.

class cepton_sdk.C_Error (code=<C_ErrorCode.CEPTON_SUCCESS: 0>, msg=None,
                        data=None)
    Error thrown by most sdk functions.

    code
        ception_sdk.C_ErrorCode

class cepton_sdk.C_Warning
```


3.1 Types

```
class cepton_sdk.ControlFlag
    An enumeration.

    DISABLE_DISTANCE_CLIP = 8
    DISABLE_IMAGE_CLIP = 4
    DISABLE_NETWORK = 2
    ENABLE_CROSSTALK_FILTER = 128
    ENABLE_MULTIPLE_RETURNS = 16
    ENABLE_STRAY_FILTER = 32
    HOST_TIMESTAMPS = 64
```

3.2 Methods

```
ception_sdk.initialize(capture_path=None, capture_seek=0, control_flags=0, error_callback=None,
                      port=None, **kwargs)
    Initializes SDK. Optionally starts capture replay.

    Parameters control_flags – ception_sdk.ControlFlag

ception_sdk.deinitialize()
```


GENERAL

API for code that is agnostic to live/replay mode.

`cepton_sdk.get_time()`

Returns capture replay time or live time.

`cepton_sdk.get_timestamp()`

Returns unix timestamp

`cepton_sdk.is_end()`

Returns true if next call to *wait* will throw *CEPTON_ERROR_EOF*

`cepton_sdk.is_live()`

Returns true if capture replay is not open.

`cepton_sdk.is_realtime()`

Returns true if live or capture replay is running.

`cepton_sdk.wait(duration=-1)`

Resumes capture replay or sleeps for duration.

If *duration* is 0, then waits forever.

SENSORS

5.1 Types

```
class cepton_sdk.SensorModel
    An enumeration.

    FUSION_790 = 8
    HR80M = 2
    HR80T = 1
    HR80T_R2 = 6
    HR80W = 3
    SORA_200 = 4
    SORA_P60 = 11
    VISTA_860 = 5
    VISTA_860_GEN2 = 7
    VISTA_M = 9
    VISTA_P60 = 12
    VISTA_X = 10
    VISTA_X15 = 13

class cepton_sdk.SensorInformation

    handle
    serial_number
    model_name
    model

    Type ception_sdk.SensorModel

    firmware_version
    formal_firmware_version
    last_reported_temperature
    last_reported_humidity
```

```
last_reported_age
last_reported_hv
last_reported_optic_temperature
gps_ts_year
gps_ts_mont
gps_ts_day
gps_ts_hour
gps_ts_min
gps_ts_sec
return_count
is_mocked
is_pps_connected
is_nmea_connected
is_calibrated
is_over_heated
```

```
class cepton_sdk.Sensor(sensor_info)
```

```
information
    Type cepton_sdk.SensorInformation
classmethod create(serial_number)
classmethod create_by_handle(sensor_handle)
classmethod create_by_index(sensor_index)
property handle
property serial_number
update()
    Update sensor information.
    Should be called often, to pull latest sensor information.
```

5.2 Methods

```
cepton_sdk.has_sensor(sensor_serial_number)
```

```
cepton_sdk.get_sensors(cls=<class 'cepton_sdk.api.Sensor'>)
    Returns attached sensors.
```

Returns Dictionary of sensors, indexed by serial number.

6.1 Types

class `cepton_sdk.Points` (*n=0*)
3D points array.

timestamps_usec

timestamps

image_positions

distances

positions

intensities

return_strongest

return_farthest

valid

saturated

`cepton_sdk.combine_points` (*points_list*)
Combine list of points (*ImagePoints*, *Points*, etc).

List must be nonempty. :returns: combined_points

All point array classes support numpy indexing and assignment as if they were 1-d arrays:

```
1 n_points = len(points_1)
2 points_2[10:20] = points_1[:10]
```

Multiple point arrays can also be combined:

```
1 points = cepton_sdk.combine_points([points_1, points_2])
```

6.2 Methods

See *Listen*.

The following methods return points directly from the C SDK callback.

`cepton_sdk.listen_frames (callback)`

Register frames callback.

Throws error if *callback_id* is currently registered.

Returns *callback_id*

`cepton_sdk.unlisten_frames (callback_id)`

Unregisters frames callback.

Throws error if *callback_id* is not currently registered.

There are also listener classes that seamlessly handle accumulation and waiting.

class `cepton_sdk.FramesListener`

Listener for getting all sensor frames.

class `cepton_sdk.SensorFramesListener (serial_number)`

6.3 Export

`cepton_sdk.export.save_points_las (points, path)`

Save points to LAS file.

`cepton_sdk.export.load_points_las (load_path, cls=<class 'cepton_sdk.point.Points'>)`

Load points from LAS file.

Returns *Points*, *extra_data*

`cepton_sdk.export.save_points_ply (points, path)`

Save points to PLY file.

`cepton_sdk.export.save_points_pcd (points, path)`

Save points to PCD file.

7.1 Methods

The following methods return serial lines directly from the C SDK callback.

`cepton_sdk.listen_serial_lines(callback)`

`cepton_sdk.unlisten_serial_lines(callback_id)`

There is also a listener class that seamlessly handle accumulation.

class `cepton_sdk.SerialLinesListener`

CAPTURE REPLAY

To open/close capture files, use `cepton_sdk.initialize` and `cepton_sdk.deinitialize` methods respectively. The high level API methods will automatically resume the capture replay as necessary.

```
cepton_sdk.open_replay(capture_path, capture_seek=0, enable_loop=False)
```

```
cepton_sdk.close_replay()
```

```
cepton_sdk.capture_replay.get_filename()
```

```
cepton_sdk.capture_replay.get_length()
```

```
cepton_sdk.capture_replay.get_position()
```

```
cepton_sdk.capture_replay.get_start_time()
```

```
cepton_sdk.capture_replay.get_time()
```

```
cepton_sdk.capture_replay.is_end()
```

```
cepton_sdk.capture_replay.is_open()
```

```
cepton_sdk.capture_replay.seek(t)
```

```
cepton_sdk.capture_replay.seek_relative(t)
```


EXPORT

Methods to import/export points to common file formats.

class cepton_sdk.export.**PointsFileType**

An enumeration.

CSV = 1

LAS = 2

PCD = 3

PLY = 4

cepton_sdk.export.**save_points** (*points, path, file_type=<PointsFileType.LAS: 2>*)

Save points to file.

Sets file extension based on type.

cepton_sdk.export.**load_points** (*path, file_type=None*)

Load points from file.

File type is inferred from extension.

Returns Points, extra_data

10.1 Multiple Sensors

Listing 1: samples/multiple_sensors.py

```
1  #!/usr/bin/env python3
2  """
3  Sample script for getting points from multiple sensors simultaneously.
4  """
5
6  import pprint
7
8  import cepton_sdk
9  import cepton_sdk.plot
10 from common import *
11
12 if __name__ == "__main__":
13     # Variables
14     capture_path = get_sample_capture_path()
15
16     # Initialize
17     cepton_sdk.initialize(capture_path=capture_path)
18
19     # Get sensors
20     sensors_dict = cepton_sdk.get_sensors()
21
22     # Get points
23     listener = cepton_sdk.FramesListener()
24     points_dict = listener.get_points()
25     del listener
26     points_list = next(iter(points_dict.values()))
27     points = points_list[0]
28
29     # Plot
30     cepton_sdk.plot.plot_points(points)
```

10.2 Single Sensor

Listing 2: samples/single_sensor.py

```
1  #!/usr/bin/env python3
2  """
3  Sample script for getting points from a single sensor.
4  """
5
6  import pprint
7
8  import numpy
9
10 import cepton_sdk
11 import cepton_sdk.plot
12 from common import *
13
14 if __name__ == "__main__":
15     # Variables
16     capture_path = get_sample_capture_path()
17
18     # Initialize
19     cepton_sdk.initialize(capture_path=capture_path)
20
21     # Get sensor
22     sensor = cepton_sdk.Sensor.create_by_index(0)
23     pprint.pprint(sensor.information.to_dict())
24
25     # Get points
26     listener = cepton_sdk.SensorFramesListener(sensor.serial_number)
27     points_list = listener.get_points()
28     del listener
29     points = points_list[0]
30
31     # Plot
32     cepton_sdk.plot.plot_points(points)
```

10.3 Advanced

10.3.1 Listen

Listing 3: samples/advanced/listen.py

```
1  #!/usr/bin/env python3
2  """
3  Sample script for the different methods of getting points.
4  """
5
6  import numpy
7
8  import cepton_sdk
9  from common import *
10
11
12 def on_frame(serial_number, points):
13     print("Received {} points from sensor {}".format(
```

(continues on next page)

(continued from previous page)

```
14         len(points), serial_number))
15
16
17 if __name__ == "__main__":
18     # Initialize
19     cepton_sdk.initialize(capture_path=get_sample_capture_path())
20     sensors_dict = cepton_sdk.get_sensors()
21     sensor = next(iter(sensors_dict.values()))
22
23     callback_id = cepton_sdk.listen_frames(on_frame)
24     cepton_sdk.wait(0.1)
25     cepton_sdk.unlisten_frames(callback_id)
26
27     # Get next frames for all sensors. Wait until data is available.
28     listener = cepton_sdk.FramesListener()
29     points_dict = listener.get_points()
30     del listener
31
32     # Get next frames for single sensor. Wait until data is available.
33     listener = cepton_sdk.SensorFramesListener(sensor.serial_number)
34     points_list = listener.get_points()
35     del listener
36
37     # Get large chunk of data
38     listener = cepton_sdk.FramesListener()
39     cepton_sdk.wait(10)
40     points_dict = listener.get_points()
41     del listener
42     points = cepton_sdk.combine_points(points_dict[sensor.serial_number])
43     print("Received {} seconds of data from sensor {}".format(
44         numpy.ptp(points.timestamps), sensor.serial_number))
```


C

`C_Error` (class in `cepton_sdk`), 3
`C_ErrorCode` (class in `cepton_sdk`), 3
`C_Warning` (class in `cepton_sdk`), 3
`close_replay()` (in module `cepton_sdk`), 15
`code` (`cepton_sdk.C_Error` attribute), 3
`combine_points()` (in module `cepton_sdk`), 11
`ControlFlag` (class in `cepton_sdk`), 5
`create()` (`cepton_sdk.Sensor` class method), 10
`create_by_handle()` (`cepton_sdk.Sensor` class method), 10
`create_by_index()` (`cepton_sdk.Sensor` class method), 10
`CSV` (`cepton_sdk.export.PointsFileType` attribute), 17

D

`deinitialize()` (in module `cepton_sdk`), 5
`DISABLE_DISTANCE_CLIP` (`cepton_sdk.ControlFlag` attribute), 5
`DISABLE_IMAGE_CLIP` (`cepton_sdk.ControlFlag` attribute), 5
`DISABLE_NETWORK` (`cepton_sdk.ControlFlag` attribute), 5
`distances` (`cepton_sdk.Points` attribute), 11

E

`ENABLE_CROSSTALK_FILTER` (`cepton_sdk.ControlFlag` attribute), 5
`ENABLE_MULTIPLE_RETURNS` (`cepton_sdk.ControlFlag` attribute), 5
`ENABLE_STRAY_FILTER` (`cepton_sdk.ControlFlag` attribute), 5

F

`firmware_version` (`cepton_sdk.SensorInformation` attribute), 9
`formal_firmware_version` (`cepton_sdk.SensorInformation` attribute), 9
`FramesListener` (class in `cepton_sdk`), 12
`FUSION_790` (`cepton_sdk.SensorModel` attribute), 9

G

`get_filename()` (in module `cepton_sdk.capture_replay`), 15
`get_length()` (in module `cepton_sdk.capture_replay`), 15
`get_position()` (in module `cepton_sdk.capture_replay`), 15
`get_sensors()` (in module `cepton_sdk`), 10
`get_start_time()` (in module `cepton_sdk.capture_replay`), 15
`get_time()` (in module `cepton_sdk`), 7
`get_time()` (in module `cepton_sdk.capture_replay`), 15
`get_timestamp()` (in module `cepton_sdk`), 7
`gps_ts_day` (`cepton_sdk.SensorInformation` attribute), 10
`gps_ts_hour` (`cepton_sdk.SensorInformation` attribute), 10
`gps_ts_min` (`cepton_sdk.SensorInformation` attribute), 10
`gps_ts_mont` (`cepton_sdk.SensorInformation` attribute), 10
`gps_ts_sec` (`cepton_sdk.SensorInformation` attribute), 10
`gps_ts_year` (`cepton_sdk.SensorInformation` attribute), 10

H

`handle` (`cepton_sdk.SensorInformation` attribute), 9
`handle()` (`cepton_sdk.Sensor` property), 10
`has_sensor()` (in module `cepton_sdk`), 10
`HOST_TIMESTAMPS` (`cepton_sdk.ControlFlag` attribute), 5
`HR80M` (`cepton_sdk.SensorModel` attribute), 9
`HR80T` (`cepton_sdk.SensorModel` attribute), 9
`HR80T_R2` (`cepton_sdk.SensorModel` attribute), 9
`HR80W` (`cepton_sdk.SensorModel` attribute), 9

I

`image_positions` (`cepton_sdk.Points` attribute), 11
`information` (`cepton_sdk.Sensor` attribute), 10
`initialize()` (in module `cepton_sdk`), 5

intensities (*cepton_sdk.Points* attribute), 11
is_calibrated (*cepton_sdk.SensorInformation* attribute), 10
is_end() (in module *cepton_sdk*), 7
is_end() (in module *cepton_sdk.capture_replay*), 15
is_live() (in module *cepton_sdk*), 7
is_mocked (*cepton_sdk.SensorInformation* attribute), 10
is_nmea_connected (*cepton_sdk.SensorInformation* attribute), 10
is_open() (in module *cepton_sdk.capture_replay*), 15
is_over_heated (*cepton_sdk.SensorInformation* attribute), 10
is_pps_connected (*cepton_sdk.SensorInformation* attribute), 10
is_realtime() (in module *cepton_sdk*), 7

L

LAS (*cepton_sdk.export.PointsFileType* attribute), 17
last_reported_age (*cepton_sdk.SensorInformation* attribute), 9
last_reported_humidity (*cepton_sdk.SensorInformation* attribute), 9
last_reported_hv (*cepton_sdk.SensorInformation* attribute), 10
last_reported_optic_temperature (*cepton_sdk.SensorInformation* attribute), 10
last_reported_temperature (*cepton_sdk.SensorInformation* attribute), 9
listen_frames() (in module *cepton_sdk*), 11
listen_serial_lines() (in module *cepton_sdk*), 13
load_points() (in module *cepton_sdk.export*), 17
load_points_las() (in module *cepton_sdk.export*), 12

M

model (*cepton_sdk.SensorInformation* attribute), 9
model_name (*cepton_sdk.SensorInformation* attribute), 9

O

open_replay() (in module *cepton_sdk*), 15

P

PCD (*cepton_sdk.export.PointsFileType* attribute), 17
PLY (*cepton_sdk.export.PointsFileType* attribute), 17
Points (class in *cepton_sdk*), 11
PointsFileType (class in *cepton_sdk.export*), 17
positions (*cepton_sdk.Points* attribute), 11

R

return_count (*cepton_sdk.SensorInformation* attribute), 10

return_farthest (*cepton_sdk.Points* attribute), 11
return_strongest (*cepton_sdk.Points* attribute), 11

S

saturated (*cepton_sdk.Points* attribute), 11
save_points() (in module *cepton_sdk.export*), 17
save_points_las() (in module *cepton_sdk.export*), 12
save_points_pcd() (in module *cepton_sdk.export*), 12
save_points_ply() (in module *cepton_sdk.export*), 12
seek() (in module *cepton_sdk.capture_replay*), 15
seek_relative() (in module *cepton_sdk.capture_replay*), 15
Sensor (class in *cepton_sdk*), 10
SensorFramesListener (class in *cepton_sdk*), 12
SensorInformation (class in *cepton_sdk*), 9
SensorModel (class in *cepton_sdk*), 9
serial_number (*cepton_sdk.SensorInformation* attribute), 9
serial_number() (*cepton_sdk.Sensor* property), 10
SerialLinesListener (class in *cepton_sdk*), 13
SORA_200 (*cepton_sdk.SensorModel* attribute), 9
SORA_P60 (*cepton_sdk.SensorModel* attribute), 9

T

timestamps (*cepton_sdk.Points* attribute), 11
timestamps_usec (*cepton_sdk.Points* attribute), 11

U

unlisten_frames() (in module *cepton_sdk*), 12
unlisten_serial_lines() (in module *cepton_sdk*), 13
update() (*cepton_sdk.Sensor* method), 10

V

valid (*cepton_sdk.Points* attribute), 11
VISTA_860 (*cepton_sdk.SensorModel* attribute), 9
VISTA_860_GEN2 (*cepton_sdk.SensorModel* attribute), 9
VISTA_M (*cepton_sdk.SensorModel* attribute), 9
VISTA_P60 (*cepton_sdk.SensorModel* attribute), 9
VISTA_X (*cepton_sdk.SensorModel* attribute), 9
VISTA_X15 (*cepton_sdk.SensorModel* attribute), 9

W

wait() (in module *cepton_sdk*), 7